

IN THE SPECIFICATION:

Please amend the paragraph beginning on page 3, line 19:

--Currently, the Web services landscape is a an evolving collection of inter-related standards and implementations. Presently, there is no system for aggregating all of the necessary information to fully describe, deploy and manage the life cycle of a Web service. Web services description language (WSDL) is an XML-based language that is central to the future development of the Web services paradigm. WSDL is used to describe the services a business offers and to provide a way for others to access those services via the Web (or any other network). The UDDI initiative is an XML-based registry standard by which businesses list themselves and the Web services they offer on the Internet. WSDL is one approach to describing such Web services. A key goal of the UDDI initiative is to enable companies to find each other and each other's Web services on the Internet and to make their computer systems inter-operable with each other in order to facilitate electronic commerce. The UDDI initiative allows businesses to list information about themselves, e.g., name, location, and/or the Web services they offer. --

Please amend the paragraph beginning on page 7, line 9:

--The invention enables one server with a Web services container to send Web services software to another server with a Web services container in order to allow that other server to begin providing that service. This may be useful, for instance, when the work load at a first server exceeds that server's capabilities. That server can then send the Web service software to one or more other servers and then divide the servicing of requests for that service amount two or more servers. The first server, for instance, can reconfigure itself either partially or totally as a "service router" to route requests for given Web services to other servers that it has determined can provide that service either by virtue of it having itself ~~sent~~ send the Web service software to the other

server(s) or by querying the other server(s) as to the contents of their Web service containers.--

Please amend the paragraph beginning on page 12, line 4:

--Figure 2 is a Universal Markup Language (UML) flow diagram illustrating container discovery according to the first method discussed above, i.e., through a registry such as UDDI. In the first step, the container 203 running on a server 201 issues a query 207 across the network to a UDDI server 205 seeking tModels consistent with the tModel of its peer containers. The UDDI directory returns to the requesting server 201 a list of tModel binding instances corresponding to the peer containers of container ~~204~~ 203 in accordance with the specified tModel. It will be understood by those of skill in the art that in the standard protocol for UML diagrams such as those shown in Figures 2 and 3, responses to queries are assumed but not shown. Only the queries are shown. More particularly, the UDDI directory returns a WSDL document containing a set of objects, the objects being the definites of the peer containers.--

Please amend the paragraph beginning on page 16, line 7:

--In addition to managing multiple services, a container also may manage multiple network access points, e.g. an HTTP access point, an HTTPS access point, an MQ access point, an SMTP access point, etc. Typically, WSDL documents comprise two parts, an abstract interface document and an implementation document. The interface document declares what the service can do (e.g., convert dollars to euros or I am a container in accordance with a certain tModel). The implementation document declares an implementation. This document ties the network access points to a particular service interface. For example, the implementation document discloses that network access point <http://www.ibm.com/smalltalk/webservice/currencyconverter:80> [http:// www .ibm.com/smalltalk/webservice/currencyconverter:80](http://www.ibm.com/smalltalk/webservice/currencyconverter:80) is bound to the

abstract interface described in the interface document. Therefore, a physical network access point may be bound to multiple services. Likewise, a service can be bound to multiple network endpoints.--

Please amend the paragraph beginning on page 21, line 2:

--Edge servers reside at the point-of-demand on the network. That is, an edge server is the server that receives a client request. Edge servers typically are used for caching static pages and performing work load management. However, applying the present invention, edge servers can be used to dynamically reconfigure the service environment. Let us assume a network topology as shown in Figure 4A, in which an edge server 212 running a Web service container 214 in accordance with the present invention acts as the front end for a back end business server 216, also running a Web service container 218 in accordance with the present invention. Edge server 212 receives a client request 226a that requires use of a currency conversion Web service 224 resident on the business server 216. The edge server 212 asks its container 214 to handle the request. Let us assume that container 214 has no services deployed, but is aware of the services available on other servers, including, at least, the business application server 216. Alternately, container 214 may actually have the currency conversion software deployed. However, let us assume that container 214 has been programmed with a predetermined threshold of Web services requests per hour it should handle and that the threshold has been exceeded.--

Please amend the paragraph beginning on page 23, line 3:

--However, if the threshold is exceeded, from step 513, the next step would be step 523, in which SOAP server 501 creates a remote message handler 525. Then, in step 527, remote message handler 525 consults the container's peer list 503 to obtain the list of peer containers. In step 529, remote message handler 525 consults the server context 505 of the peers listed in peer list 503 to determine if any one of them

can handle this particular message. Let us assume for sake of illustrating even further features of the invention, that the server context information 505 for one or more particular peer containers contains insufficient information to determine whether it can handle the message. Accordingly, in step 531, the server context 505 is sent in message 531 over the network to the SOAP server 507 of the particular peer container. SOAP server 507 consults its own server context 509 to get the additional needed information and returns that additional contextual information (as shown in step 535) to the SOAP server 507. Although not shown in Figure 5, that additional contextual information is further returned from SOAP server 507 to server context 505 and then on to the remote message handler 525.--

Please amend the paragraph beginning on page 30, line 15:

-- Consider the following example, which helps illustrate two features of the present invention, namely, a client machine running a Web service container and platform independent service delivery and distribution. Referring to Figure 4D, a client machine 230 and an application server 232 are in communication via the Internet 14. Let us assume that the client machine 230 is a personal digital assistant coupled to the Internet, such as a wireless ~~Palm Pilot™~~ PALM PILOT™. Each has a Web service container 234 and 236, respectively. A currency conversion Web service 240 has been deployed in the application server container 236 that converts between US dollars and euros using current exchange rates.

Please amend the paragraph beginning on page 31, line 4:

--The business application service in ~~Palm Pilot™~~ PALM PILOT™ 230 can instruct the ~~Palm Pilot™~~ PALM PILOT™ container 234 to issue a request 238a to the posting Web application server container 236 for a copy of the currency conversion Web service 240. Container 234 can include in the header of that request information about its local container. For instance, it might include in a first request for a Web

service the information that the requesting device is a ~~Palm Pilot™~~ PALM PILOT™ with a Java JAVA stack. The Web application server container 236 examines the request and recognizes that the container 234 that issued the request has identified that the device on which it is implemented is a ~~Palm Pilot™~~ PALM PILOT™ device. Let us also assume that the Web application server container 236 has available to it several implementations of the currency conversion Web service software for various platforms, including, for example, the ~~Palm™~~ PALM™ platform, a Java™ JAVA™ platform and Microsoft's MICROSOFT™s .Net platform. Thus, the server can issue a response 238b to the request informing the client container of this option. The client container can then choose to download the software or simply have it serviced by the server in the normal fashion. Referring now to Figure 4E, if the client issues a request 238c to download the software module, the Web application server container 236 issues a response 238d forwarding a ~~Palm™~~ PALM™ version of the currency conversion software to the palm container. Upon receipt, the palm container 234 loads the code and dispatches all future requests to the local service 240'. Accordingly, Web service software can be exchanged between network nodes running different software and/or hardware platforms. This concept is termed "location transparency" since the real location and implementation are "transparent" to the ultimate consumer of the service. In addition, this implies a conceptual abstraction of the code representation of a Web service. That is, it may be acceptable for a service to have multiple implementations (e.g. one for cell phones and another for ~~Palm~~ PALM™ devices) that are hidden within a container.

Please amend the paragraph on page 32, line 8:

--The scenario described above can also be applied to provide game software to requesting clients. All that is required is that the requesting device, e.g., a ~~Sony PlayStation 2™~~ SONY PLAYSTATION 2™, implement a service container in accordance with the present invention. Games can then be distributed to both PCs and

set top boxes automatically.--

Please amend the paragraph beginning on page 32, line 12:

--When a client issues an original request for a Web service, it can include in the header of that request information about its local container. For instance, a ~~Palm Pilot~~TM PALM PILOTTM might include in a first request for a Web service the information that it is a ~~Palm Pilot~~TM PALM PILOTTM with Java JAVATM runtime capability. Let us assume that the server that receives the request has a copy of the Web service software module in the ~~Palm Pilot's~~ PALM PILOTTM's platform that the client can download from the server and thereafter use locally, rather than going over the network for the Web service. Thus, the server can respond to the request informing the client container of this option. The client container can then choose to download the software or simply have it serviced by the server in the normal fashion.--

Please amend the paragraph beginning on page 33, line 9:

In even further embodiments, Web service containers in accordance with the present invention may provide automatic micro-payment functionalities. Merely by way of example and not limiting, (1) clients can be charged periodic subscription charges, (2) clients can be charged on a per use basis for each invocation of the service, (3) clients may lease a service (for example, by purchasing a number of invocations or usage for a specified time period), (4) clients may directly purchase and upload a service from a container, and (5) clients may be allowed to use a service for a fixed number of invocations or a fixed time period, after which they must choose to purchase.